Paper 306-2013

Accessing SAS® Code via Visual Basic for Applications

Jennifer Davies, Z, Inc, Silver Spring, MD

ABSTRACT

SAS® software has functionality that applications such as Microsoft Access or Excel do not have and vice versa. However, in some situations, Microsoft applications are preferred by the user over SAS for a multitude of reasons. This paper will discuss how to integrate the use of Microsoft applications with the functionality of SAS programs. This becomes very important when SAS Business Intelligence is not available.

Depending on how SAS is installed in the user's organization, the programmer may have to access SAS on the PC or a server version of the application. This paper will explain the two methods used for calling SAS code from Visual Basic for Applications (VBA) Code (v6.5).

INTRODUCTION

Oftentimes, applications need to be setup for users who do not have the extensive technical skills and depend on programmers to enable them to do their work. In the situations discussed in this paper, the users had specific reasons for keeping the bulk of the work in a Microsoft application. However, the applications used could not perform the heavy analysis done in SAS as easily. For this reason, SAS is called in the background to run a program and return the results for the application to use. The Visual Basic for Applications code illustrated here will work in either Microsoft Access or Excel. The first example will be calling PC SAS and the second will be for server based SAS.

CALLING PC SAS FROM VISUAL BASIC FOR APPLICATIONS

In this situation, the user has both Microsoft Access 2007 and Base SAS 9.1 installed on the PC. The Microsoft Access database application receives data from an outside source, loads it into the database, and then runs some estimations and forecasting in SAS on the newly received data. The results are loaded into an Oracle table, which are then read by the user in the Microsoft Access database. The user of this database has very little technical knowledge so all the processing is done with a series of VBA macros that are enabled on a user interface.

The code below is the full VBA code used to connect to Base SAS. As SAS programmers, the syntax may be unfamiliar so I will explain each line of the code. The code is using macro variables and lookup functions to query a lookup table that enables the user to make changes to the paths, etc without a programmer needing to alter the hard coded entries in the code.

```
Sub Launch__Forecast_model_in_SAS() ← the start of a code module in VBA
```

```
  MsgBox "Now launching Forecast model in SAS" ← Messages the user what function the VBA code is now starting.
```

```
    SAS_installation_folder = DFirst("SAS_installation_folder", "Miscellaneous_defaults") ← establishes the path where Base SAS is installed. SAS_installation_folder is a variable on the Miscellaneous_defaults. This resolves to: C:\PROGRAM FILES\SAS\SASFOUNDATION\9.2
```

```
    SAS_Code_folder = DFirst("SAS_Code_folder", "Miscellaneous_defaults")← establishes the path where the SAS program is stored. SAS_installation_folder is a variable on the Miscellaneous_defaults. This resolves to: X:
```

```
  shell_command = SAS_installation_folder & "\SAS.EXE -SYSIN " &
SAS_Code_folder & "\FORECAST.SAS" ← establishes a shell command which opens Base SAS and
runs the specified SAS program.  The full shell command would read: C:\PROGRAM
FILES\SAS\SASFOUNDATION\9.2\SAS.EXE –SYSIN X:\FORECAST.SAS
    retval = Shell(shell_command, 1)  ← Runs the shell command established above.

  Exit Sub ← Exits the sub module

End Sub ← Ends the VBA module
```

As you can see from this code, there is no need for the user to login to the system. The SAS code does contain a prompt to login for the Oracle connection, but there is no login for the SAS connection. In the next example, a login for the SAS server will be included.

USING VISUAL BASIC FOR APPLICATIONS TO RUN A SAS PROGRAM ON THE SAS SERVER

In the following example, the user has been upgraded from using Base SAS 9.1 on the PC to SAS Enterprise Guide® 4.3 with a remote server connection to a configured SAS Metadata server and Workspace server. These SAS servers are installed and configured as part of the SAS Enterprise Business Intelligence Server solution on the UNIX Server.

The users now have to make a connection to the SAS server and be authenticated with a user name and password. Some of the users are enabled to make Unix X-Term commands and will be using a different port and workspace name than the rest of the users. Again, to enable the authentication process and differing ports, macro variables are established to enable the users to change the information rather than hard coding the statements. This example will run the same SAS program as the example above, using the same Microsoft Access application.

SETUP

In order to establish a connection to the SAS server, certain references will need to be installed and enabled on each PC running the code. If PC SAS is no longer installed on the PC, the users can use SAS Integrated Technologies to install the references. After the installation, the plug-ins will then need to be enabled in VBA (Microsoft VBA → Tools → References).

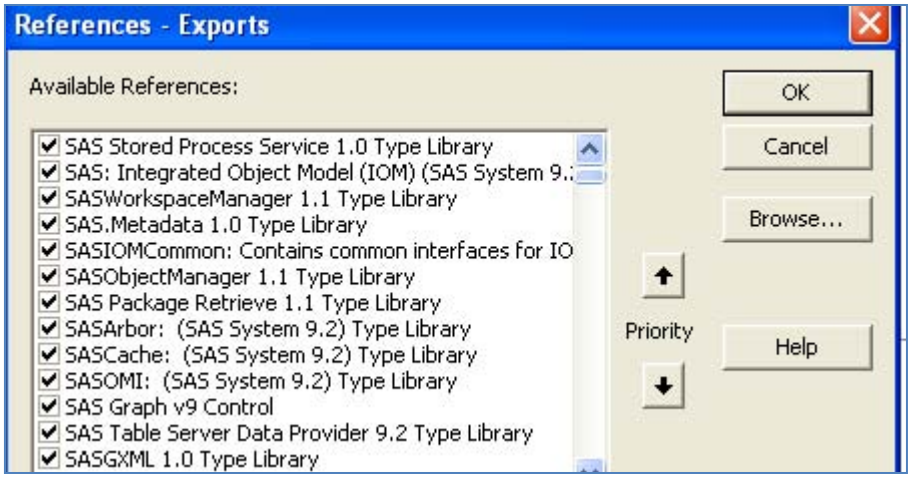

Fig. 1

The easiest way to handle the authentication into the SAS server is to prompt the user for credentials. Based on the answers to the prompts, the VBA code will supply the port and workspace name to the macros. The simple screen below requests the user id and password, encrypting the password with asterisks.

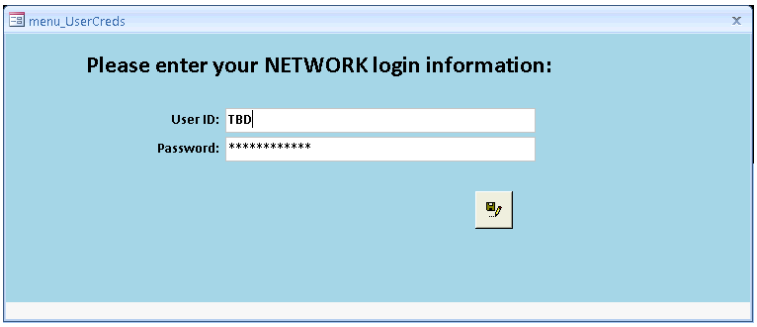

Fig. 2

Each user will have their user id, port and workspace name entered in a lookup table that the VBA code will reference. User authentication can be handled with various methods. The entry screen shown above (fig. 2) is an easy way to automate the entry of the user authentication information. The port and workspace name is stored within a lookup table and extracted using the DLOOKUP function. When using this connection in Microsoft Excel, a similar form can be used for input with the difference being that the port and workspace name is stored on a hidden sheet instead of a table.

VISUAL BASIC FOR APPLICATION CODE

Once the entries for authentication are established, the connection to the SAS server can begin. The VBA code below connects to the server and runs the same SAS code that the first example ran. Each line of the code will be explained. The code below will use an IOM Bridge to connect to the SAS server and LanguageService to submit the code. There are other connections available depending on the user's organization but this was the best choice for this situation.

```
Sub Launch__Forecast_model_in_SAS_new() ← Begins the VBA code module


    'request and populate the user credentials ← Code comment
  DoCmd.RunMacro "macro_usercreds" ← Launches the credentials screen shown in Fig. 1.


  Dim userid As String
    Dim passid As String
    Dim portid As String
    Dim nameid As String
```

Defines the variables used in the Visual Basic code as string (character) variables

```
  userid = DLookup("userid", "tbl_usercreds", "userid is not null") ← Verifies that the userid is in the lookup table tbl_usercreds

  passid = DLookup("pass", "tbl_usercreds", "pass is not null") ← Verifies that the password is in the lookup table tbl_usercreds

  portid = DLookup("port", "tbl_userports", "[userid]= " & """" & userid & """") ← Returns the port based on the userid when searched in the table tbl_userports

    nameid = DLookup("portname", "tbl_userports", "[userid]= " & """" & userid & """") ← Returns the workspace name based on the userid when searched in the table tbl_userports
```

```
  MsgBox "Now launching Forecast model in SAS to Load to ORA2A"
```
← Messages the user that SAS code is being launched and where the results will be loaded.

```
    Dim obObjectFactory As New SASObjectManager.ObjectFactory

    Dim obObjectKeeper As New SASObjectManager.ObjectKeeper

    Dim obSAS As SAS.Workspace

    Dim cn As New ADODB.Connection

    Dim rs As New ADODB.Recordset

    Dim observer As New SASObjectManager.ServerDef
```

Defines the variables used in the Visual Basic code to access the various plug-ins required for access to the server.

```
  observer.MachineDNSName = "sasserver1.machinename.path"
```
← Define the path of the server

```
  observer.Port = portid
```
← Based on the lookup returned above (ie. 8645)

```
  observer.Protocol = ProtocolBridge
```
←Default protocol

```
  observer.NAME = nameid
```
← Based on the lookup returned above (ie. SASApp - Workspace Server)

```
    Set obSAS = obObjectFactory.CreateObjectByServer("sasserver1", True, observer, userid, passid)
```
← Establishes the connection to the SAS server with login parameters within the parenthesis.

```
  obSAS.LanguageService.Submit ("options source2; %include
'/sasserver/share_folder/Forecast/Forecast.sas';")
```
← Submits the SAS program. Notice that the path for the program is a Unix path due to being stored on the server.

```
  Dim sLog As String
```
← Define variable used for logging as a string (character)

```
  sLog = obSAS.LanguageService.FlushLog(200000)
```
← Set the variable to the beginning of the SAS log

```
  MsgBox "" & sLog
```
←Display the log

```
  MsgBox "Full SAS log can be viewed at: \\sasserver1\share_folder\FORECAST"
```
← Message the user where the full log can be found.

```
     obSAS.Close
```
← Close the connection to the SAS server

```
    Exit Sub
```
← Exit the sub module

```
End Sub
```
← Exit the VBA module

By making the connection variables parameterized, the code can easily be used by multiple users. This code can also be easily used in Microsoft Excel with minor changes to the parameter definitions. The code used to connect to the SAS server and run the program will remain the same.

CONCLUSION

In my organization, the users had been connected to this specific piece of SAS code for many years with PC SAS. When our SAS was upgraded to SAS Enterprise Guide and we began connecting to a server, we had to establish a new method for connection. In researching the various ways to do this, we found many possibilities

but found that the IOM Bridge and LanguageServices were best for this. Our other consideration was the ease of use. The main users of this database are not strong programmers. Enabling user interface screens for parameterizing the authentication requirements allowed multiple users with no programmer intervention. While both methods explained in this paper will return the same results, the decision on which method to use will be determined by the organizations' requirements.

REFERENCES:

Recommended reading sources:

Ivey, William. "Using Open Access Technology to Integrate SAS and Esri Applications" - http://proceedings.esri.com/library/userconf/proc01/professional/papers/pap571/p571.htm

Pratter, Frederick. "Access to SAS® Data Using the Integrated Object Model (IOM) in Version 9.1" - http://www.lexjansen.com/pharmasug/2005/applicationsdevelopment/ad05.pdf

SAS Knowledge Base Sample 26145 "Visual Studio 2005 VBA Code Snippets" -

http://support.sas.com/kb/26/145.html

ACKNOWLEDGMENTS

In addition to the references found on the web, I would like to acknowledge the support of my co-workers Jasmin Kakar, Virginia Blum, Farhad Balashov, Jeff Greear, Scott Sheck and Van Smith in their help for squashing the bugs we encountered along the way. Also, I would like to thank Garrett and Ronnie of the SAS Technical Support for recommended tweaks that allowed this code to run smoothly.

TRADEMARKS

CONTACT INFORMATION

Contact the author at:

Jennifer Davies
Z, Inc
Email: Mail4jenz@gmail.com